XXVIII Congresso {virtual} de Iniciação Científica da Unicamp 2020

Avaliação de estratégias de busca de similaridade usando funções de pareamento aproximado para investigações forenses

Aluno: João P. B. Velho, Orientador: Marco A. A. Henriques

[1]Faculdade de Engenharia Elétrica e de Computação (FEEC)

{bizzi,marco}@dca.fee.unicamp.br

1. Introdução

O avanço tecnológico, apesar de todos os seus benefícios para a sociedade, torna as investigações no âmbito da forense digital cada vez mais desafiadoras. O aumento na quantidade e também na capacidade de armazenamento de dispositivos digitais faz com que o trabalho de análise destes dispositivos pelos peritos forenses se torne cada vez mais árduo. Normalmente, são utilizadas funções hash para gerar resumos dos arquivos de uma mídia e facilitar a localização de artefatos já conhecidos e de interesse em uma investigação, como exemplares de malware, arquivos de listas negras etc. Estas funções são conhecidas por serem eficientes e conseguirem lidar com grandes quantidades de dados em um curto período de tempo. Contudo, devido ao efeito avalanche presente nestas funções (alteração de um bit na entrada gera uma saída completamente diferente), uma limitação se torna clara neste contexto: a ineficiência para busca de similaridade.

Para solucionar o problema envolvendo a identificação de similaridade, foram desenvolvidos nos últimos anos as funções de Pareamento Aproximado (PA). Assim como as funções hash, as técnicas de PA tem como objetivo criar resumos para os arquivos e, através da simples comparação destes resumos, avaliar a similaridade de forma eficiente, na qual arquivos similares apresentam resumos similares. Além de identificar artefatos de interesse em uma investigação, estas funções também podem ser empregadas em sistemas anti-vírus para identificar malwares de forma mais eficiente que os métodos de assinatura tradicionais.

O problema que ainda permanece com as funções de PA é que, dados dois conjuntos com muitos arquivos, ainda é inviável a comparação destes conjuntos utilizando métodos convencionais como o de força bruta, onde uma comparação arquivo por arquivo entre os conjuntos é realizada. Dado a grande quantidade de dados presentes em investigações de hoje em dia, métodos simples de busca como o citado se tornam impraticáveis. Por esta razão, foram desenvolvidas na literatura as estratégias de busca de similaridade, que são formas mais eficientes da utilização das funções de PA para a realização de comparações entre grandes conjuntos de arquivos [Moia and Henriques 2017]. Utilizando os resumos de um conjunto de arquivos gerados pelas funções de PA, as estratégias criam uma estrutura de armazenamento para estes resumos, seja uma tabela ou um filtro de Bloom, por exemplo, de forma a organizar os resumos criados e permitir que buscas de similaridade entre dois conjuntos sejam realizadas de forma mais eficiente.

No entanto, as estratégias de busca de similaridade ainda carecem de uma avaliação e comparação conjuntas, analisando como tais abordagens se comportariam na prática. Deste forma, este trabalho busca avaliar e comparar algumas das estratégias existentes, como o `MRSH-NET` e o `F2S2` (escolhidas por serem as precursoras da área de PA) além de métodos de força bruta relacionados. Mostramos sob a ótica das métricas de *precision* e *recall* como estas estratégias lidam com dados reais. Também avaliamos o tempo de execução e o impacto do tipo de arquivo na similaridade. Melhorias são propostas na forma de uma modificação na estratégia `MRSH-NET`, a fim de melhorar sua capacidade de detecção de similaridade, como mostrado e comprovado em nossos experimentos, onde os resultados se mostram similares aos das demais estratégias e até superiores em alguns casos. Este trabalho está estruturado da seguinte forma: Seção 2 apresenta o problema estudado, seguido de nossa modificação do `MRSH-NET`. Os dados e métricas utilizadas no estudo são apresentados em seguida. Finalizamos com a discussão dos resultados obtidos e conclusões.

2. Descrição do Problema e Direções da Pesquisa

As funções de Pareamento Aproximado apresentam altos tempos de execução quando submetidas a grandes volumes de dados. Dado dois conjuntos, sendo o primeiro composto por arquivos de referência de posse de um investigador de tamanho r e o segundo por arquivos de interesse (a serem analisados) de tamanho q, compará-los

resulta em uma complexidade de busca de $O(qr)$. Para minimizar esta alta complexidade, foram propostas diferentes estratégias de busca de similaridade que conseguem reduzir a complexidade para até $O(q)$ em alguns casos.

As estratégias analisadas neste trabalho, isto é, o F2S2 e o MRSH-NET, apresentam particularidades em relação a forma de apresentar seus resultados, o que pode ser suficiente ou até limitar uma investigação. Por utilizar um filtro de Bloom para mapear todos os arquivos de um conjunto, o MRSH-NET retorna apenas uma resposta binária a uma consulta, sendo capaz de afirmar se um dado arquivo está presente no conjunto mapeado em sua estrutura ou não; ele não indica a qual arquivo do conjunto o item consultado é similar. Contudo, seu benefício está em sua menor complexidade: $O(q)$. Já o F2S2 é capaz de indicar a qual arquivo do conjunto mapeado o item consultado é similar. Por outro lado, sua complexidade é $O(qr)$, apesar de testes mostrarem que na prática a redução do tempo é significativa [Winter et al. 2013].

É crucial levar em consideração o tipo de resposta, assim como a complexidade de busca, no momento de escolher qual estratégia utilizar. Além disso, a capacidade de detecção também deve ser um fator indispensável para a tomada de decisão. Desta forma, a primeira contribuição deste trabalho é a proposta de uma nova versão da abordagem MRSH-NET, com melhorias focadas em diminuir a complexidade de busca e aumentar a capacidade de detecção desta estratégia.

Outra contribuição deste trabalho é a comparação das estratégias em relação à capacidade de detecção, em especial analisando as taxas de *precision* e *recall*. Até onde sabemos, nenhum trabalho na literatura realizou tal análise. São comparadas as estratégias mencionadas acima (com adição da versão que propomos) e são analisadas as taxas de *precision* e *recall*, mostrando o impacto dos diferentes tipos de arquivos na similaridade. Uma análise dos tempos de execução das estratégias também é realizado. É importante destacar que, como não existiam implementações de código-aberto da estratégia F2S2 (até onde sabemos), foi desenvolvida e disponibilizada uma prova de conceito desta estratégia em https://github.com/regras/f2s2. O código desenvolvido realiza as principais funções da estratégia: a criação da estrutura, inserção dos resumos e a consulta de um resumo na estrutura.

3. Aprimorando o MRSH-NET

A estratégia MRSH-NET utiliza a ferramenta mrsh-v2 para extração de features dos arquivos. Apesar de possuir características interessantes para investigações forenses, [Breitinger and Roussev 2014] mostram que esta ferramenta gera muitos falsos positivos (diminuindo sua taxa *precision*), além de ter um desempenho geral menor que o sdhash. Neste trabalho, decidimos realizar a substituição do módulo de extração de features do mrsh-v2 pelo módulo do sdhash, a fim de melhorar as capacidades de detecção da estratégia MRSH-NET. Batizamos a versão modificada de MRSH-SD; nas próximas seções serão discutidas as vantagens e desvantagens de nossa alteração.

É importante destacar que a mudança realizada não altera a forma de interpretação dos resultados da nova estratégia. Outro ponto a ser enfatizado é que a versão original da estratégia impõe uma condição para considerar um arquivo como similar. Esta condição requer que o arquivo consultado tenha, no mínimo, seis (6) features (em comum) consecutivas encontradas em sua estrutura durante uma busca; caso contrário, o arquivo será considerado diferente por não haver similaridade suficiente. Esta decisão foi mantida a princípio no MRSH-SD, porém testes variando este valor de inicial de *threshold* serão realizados a fim de observar o impacto nas taxas de *precision* e *recall*. Ressaltamos ainda que os parâmetros utilizados na extração e seleção de features são os mesmos utilizados pelo sdhash original. O código-fonte da implementação de nossas mudanças se encontra disponível em https://github.com/regras/mrsh-sd.

4. Dados e métricas utilizados nos experimentos

Para realizar os experimentos deste trabalho, utilizamos dois conjuntos de arquivos extraídos da base de dados *t5-corpus* [Roussev 2011]. Estes conjuntos foram denominados como *target* e *known*. Foram utilizados conjuntos relativamente pequenos para facilitar a análise manual dos resultados. Assim é possível saber quais arquivos são realmente similares para calcular as métricas *precision* e *recall*.

Além de avaliar as estratégias, adicionamos também duas abordagens que utilizam o método de força bruta usando as ferramentas de Pareamento Aproximado sdhash e ssdeep. Foi gerado um resumo para cada arquivo dos conjuntos *known* e *target* e realizada uma comparação entre os conjuntos (todos-contra-todos) para cada ferramenta a fim de determinar os pares similares entre os conjuntos. Já para as estratégias de busca de similaridade, o conjunto *known* foi mapeado na estrutura da estratégia, enquanto os arquivos do conjunto *target* tiveram seus resumos calculados e consultados um a um na estrutura criada.

Neste trabalho, avaliamos a capacidade de detecção das estratégias através de três métricas recorren-

tes no cenário de pareamento aproximado e que são utilizadas na área de *Recuperação de Informação*: *precision*, *recall* e *F-score*. A métrica *precision* avalia a proporção de acertos da estratégia, ou seja, pares de arquivos realmente similares (tp) em relação a todos os resultados retornados, incluindo falsos positivos ($tp + fp$). O *recall* se refere a proporção de arquivos similares encontrados (tp) em relação à todos os arquivos similares existentes entre os conjuntos ($tp + fn$). Já o *F-score* é a média harmônica entre as duas métricas anteriores, sendo um o melhor resultado que pode ser alcançado.

$$precision = \frac{t_p}{tp+fp}$$
$$recall = \frac{tp}{tp+fn}$$
$$F-score = 2 * \frac{precision*recall}{precision+recall}$$

Para calcular as métricas citadas, é necessário saber quando uma estratégia acerta ou erra ao comparar dois arquivos. É importante ressaltar que existem diferentes formas de similaridade. [Moia et al. 2020] define que a similaridade encontrada entre arquivos pode ser de três tipos: conteúdo gerado por usuário - *User Generated Content* (UGC), conteúdo gerado por aplicação - *Application Generated Content* (AGC) ou conteúdo gerado por template - *Template Content* (TC). Neste trabalho, adotaremos a mesma classificação de similaridade. Além do mais, consideraremos como similar, ou verdadeiro positivo (tp), apenas resultados de comparações relacionadas a UGC ou TC, visto que são as duas normalmente de maior interesse em uma investigação. Similaridades do tipo AGC serão consideradas falsos positivos (fp); verdadeiros negativos (tn) serão as similaridades não encontradas de AGC, enquanto falsos negativos (fn) são similaridades não encontradas de UGC e TC. Cenários em que os tipos de similaridades de interesse são outros (por exemplo, apenas TC e AGC), serão abordados em trabalhos futuros.

Para classificar as similaridades retornadas pelas estratégias dentre os três tipos, utilizaremos os resultados de uma classificação manual disponível em `https://github.com/regras/cbamf/blob/master/Some_matches_t5-corpus_per_similarity_class.txt`. Estes dados foram organizados em forma de lista e têm grande parte das comparações entre os conjuntos analisados que possuem algum nível de similaridade, além do seu tipo (UGC, AGC ou TC). Qualquer similaridade reportada pela ferramenta entre arquivos que não constem nesta lista será considerada como não-similar. Iremos nos referir a esta lista como `lista-verdade` .

Outro ponto a ser destacado é que, devido ao funcionamento de cada estratégia, alguns procedimentos para a avaliação mudam de uma estratégia para outra. Como a estratégia `F2S2` não indica para quais arquivos não há similaridade quando realizamos uma consulta, neste caso assumimos que todos os arquivos do conjunto *known* são diferentes e calculamos o tn e o fn nos baseando na `lista-verdade` . Já quando um arquivo é dito similar, calculamos o tp e o fp também nos baseando na `lista-verdade` mas agora para saber se houve acerto ou erro. Ressaltamos que o `F2S2` pode retornar nenhum ou vários arquivos com similaridade para cada item consultado.

Para as estratégias, `MRSH-NET` e `MRSH-SD`, o procedimento de avaliação é diferente. Como estas estratégias apenas nos fornecem uma resposta binária para cada consulta, calculamos apenas um único valor relativo ao acerto ou erro destas abordagens com base na `lista-verdade` . Assim, apenas um dos quatro possíveis valores é retornado para cada consulta: tp, fp, tn ou fn.

A última observação é em relação aos parâmetros internos utilizados pelas estratégias. O `F2S2` requer a definição do número de entradas da tabela hash a ser criada. Utilizamos três bytes do *n-gram* para criação do índice da tabela, o que leva a uma tabela com 2^{24} entradas, mais do que suficiente para a quantidade de dados disponíveis. Já para o tamanho do filtro de Bloom das estratégias `MRSH-NET` e `MRSH-SD`, utilizamos a Eq. 1 com os seguintes valores: probabilidade de ocorrência de um falso positivo $p_f = 10^{-6}$, número de subhashes $k = 5$, e número mínimo de features subsequentes em comum (para uma comparação ser considerada similar) $r = 6$. O tamanho do conjunto mapeado na estratégia é de $s = 1.7GB$. Com base nos parâmetros fornecidos, o tamanho mínimo do filtro é de 33 MB.

$$m = -\frac{k \cdot s \cdot 2^{14}}{\ln(1 - \sqrt[kr]{p_f})} \quad (1)$$

5. Resultados e Discussões

Iniciamos esta seção com os resultados da busca de similaridade utilizando as estratégias apresentadas na Sec. 2 e os conjuntos de dados da Sec. 4. Para cada estratégia, calculamos o *precision*, *recall* e *F-score*. O impacto do tipo de arquivo na similaridade também é analisado, assim como o tempo gasto na comparação dos conjuntos. No artigo publicado também é apresentado uma análise do valor *threshold (t)* de features consecutivas para a melhoria `MRSH-SD` e também uma análise dos tempos de comparação das estratégias.

5.1. Analisando a capacidade de detecção das estratégias

Nesta seção, discutimos os resultados das comparações dos conjuntos analisados pelas estratégias

de busca de similaridade. Para realização dos experimentos, foram utilizados os parâmetros padrão das estratégias MRSH-NET e F2S2, com variação apenas do tamanho de suas estruturas. Já para o MRSH-SD, utilizamos três versões, onde em cada uma alteramos o valor de *threshold (t)* de features consecutivas para representar diferentes situações que privilegiam determinada métrica. Os valores considerados foram: 6, 50 e 130. Os resultados são apresentados na Tabela 1.

Tabela 1. Resultado das comparações das estratégias de busca de similaridade e alguns métodos de força bruta para diferentes métricas

Estratégia	*Precision*	*Recall*	*F-score*
ssdeep	**0,778**	0,686	0,729
sdhash	0,249	0,983	0,397
F2S2	0,759	0,694	0,725
MRSH-NET	0,629	0,944	0,755
MRSH-SD ($t = 6$)	0,620	**1,000**	**0,765**
MRSH-SD ($t = 50$)	0,662	0,796	0,723
MRSH-SD ($t = 130$)	0,760	0,648	0,699

Podemos notar pelos resultados que diferentes estratégias se sobressaíram em diferentes métricas. A abordagem força bruta utilizando a ferramenta ssdeep obteve o melhor valor de *precision*, apresentando a menor quantidade de falsos positivos. Por outro lado, apresentou um dos piores *recall*, onde várias comparações similares não foram detectadas. Já a estratégia que conseguiu retornar todas as comparações similares foi o MRSH-SD ($t = 6$), porém devido ao baixo valor de *threshold* utilizado, apresentou vários falsos positivos; contudo, mesmo com esta limitação, obteve a melhor relação de *precision* e *recall*, representada pela métrica *F-score*. Portanto, nossa decisão em substituir a ferramenta de PA da estratégia MRSH-NET se mostrou benéfica para aumentar a capacidade de detecção de similaridade entre arquivos pela nova versão.

É importante ressaltar que os baixos valores de *precision* obtidos com todas as estratégias, principalmente com o sdhash, são devido à similaridade gerada por conteúdo de aplicação que foi considerado de não interesse em nossos experimentos. Este tipo de dado, referido na literatura como *blocos comuns* [Moia et al. 2020], será objeto de estudos futuros, onde esperamos que, após identificar e remover estes blocos, obtenhamos uma melhora significativa nos resultados de *precision* das estratégias.

Outro ponto a ser destacado em nossos resultados é em relação a capacidade de detecção de similaridade da ferramenta em comparação com a estratégia levando em conta as métricas analisadas. Gostaríamos de saber se a estratégia limita a capacidade de detecção da ferramenta por ela utilizada. Para responder a esta pergunta, comparamos as estratégias F2S2 e MRSH-SD com as respectivas ferramentas por elas utilizadas (ssdeep e sdhash) sob a abordagem força bruta. No primeiro caso, percebemos que ambas tiveram resultados semelhantes. Se por um lado ssdeep teve uma pequena vantagem em *precision*, F2S2 se saiu melhor em *recall*, porém a diferença foi mínima. Em relação ao segundo caso, podemos comparar sdhash com as três versões do MRSH-SD. A estratégia obteve resultados melhores para *precision* e *F-score* nas três versões. Além disso, na configuração com menor valor de *threshold*, a estratégia obteve também um valor maior de *recall*, onde conseguiu encontrar todas as comparações similares. A melhoria dos resultados das estratégias em relação às ferramentas pode ser explicada pela alteração do modo de avaliação da similaridade, que apesar de utilizar o mesmo processo de extração de features, mudou a forma de armazená-las e compará-las. Desta forma, podemos concluir que não há redução nas capacidades de detecção de similaridade das funções de PA por parte das estratégias (há até melhorias, em alguns casos).

5.2. Avaliando o impacto de diferentes tipos de arquivos na similaridade

Outra avaliação realizada neste trabalho foi verificar o impacto dos diferentes tipos de arquivos na detecção de similaridade pelas estratégias. Nesta análise, descartamos os métodos de força bruta pois a capacidade de detecção das estratégias é próxima (ou até superior) à deles.Novamente, será considerado que apenas similaridades entre arquivos com conteúdo gerado por usuário ou template. Além disso, serão descartadas as comparações dos tipos de arquivo gif e jpg devido ao baixo número de comparações similares das classes consideradas.

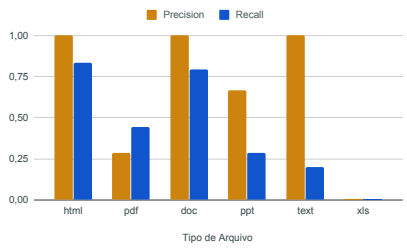

Figura 1. Taxas de *Precision* e *Recall* separadas por tipo de arquivo para a estratégia F2S2

As Figuras 1, 2 e 3 mostram os resultados das comparações das estratégias F2S2, MRSH-NET e MRSH-SD ($t = 6$), separados por tipo de arquivo. Para o cenário considerado em nossos experimentos, notamos que a taxa de *precision* se mantém alta para a maioria dos tipos de arquivos para a estratégia F2S2. A exceção são os arquivos do tipo pdf, com muitos casos de falsos positivos (similaridade por AGC), em razão dos *blo-*

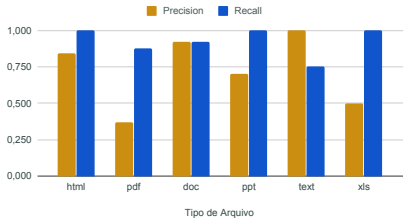

Figura 2. Taxas de *Precision* e *Recall* separadas por tipo de arquivo para a estratégia `MRSH-NET`

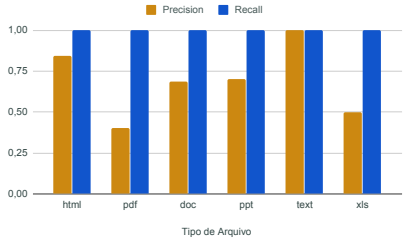

Figura 3. Taxas de *Precision* e *Recall* separadas por tipo de arquivo para a estratégia $\texttt{MRSH-SD}_6$

cos comuns. Já em relação a *recall*, o `F2S2` não conseguiu encontrar nenhuma das seis comparações do tipo `xls` existentes, mas também não acusou nenhum falso positivo (não retornou nenhuma comparação deste tipo). O baixo nível de *recall* pode ser explicado pela ferramenta utilizada (`ssdeep`), sendo esta vulnerável a modificações aleatórias em arquivos, como no caso de comparações do tipo `text` e `xls`. Este último é desafiador devido à organização dos dados, que podem estar em diferentes abas, linhas e colunas.

Já o `MRSH-NET` obteve altas taxas de *precision* e *recall* para arquivos `html`, `doc` e `ppt`. Porém, uma razoável quantidade de falsos positivos gerados por arquivos `pdf` e `xls` foi observada novamente, apesar de esta estratégia já ter identificado todas as similaridades existentes. Destacamos uma melhora em *recall* para arquivos `text` em relação à estratégia anterior, porém ainda não satisfatória. Aqui fica claro que a mudança da ferramenta usada pelas estratégias (de `ssdeep` em `F2S2` para `mrsh-v2` em `MRSH-NET`) fez a diferença no quesito *recall*.

Por fim, analisamos os resultados do `MRSH-SD` ($t = 6$) e novamente vemos o impacto da troca da ferramenta, agora usando o `sdhash` ao invés do `mrsh-v2`. Esta escolha, aliada a um bom valor de *threshold* (seis, na versão considerada aqui), fez com que a estratégia encontrasse todos os itens similares; porém, isto reflete em uma pequena redução em *precision*, em particular para arquivos do tipo `doc` e `ppt`. Tal redução se dá devido à presença dos *blocos comuns* gerados por aplicações, que são encontrados em maiores quantidades justamente nestes tipos de arquivos.

Observando os três gráficos apresentados, nota-se que a maior dificuldade das estratégias foi com os arquivos do tipo `pdf`, que geram grandes quantidades de falsos positivos. O mesmo se repetiu para arquivos do tipo `xls`, onde o `F2S2` não conseguiu encontrar nenhuma comparação similar e as estratégias `MRSH-NET` e `MRSH-SD` ($t = 6$) apresentaram altas taxas de falsos positivos devido a similaridades AGC. As informações similares neste tipo de arquivo são fragmentadas facilmente, o que torna o trabalho das ferramentas mais desafiador. Ressaltamos também que todas as estratégias se saíram bem buscando similaridades de arquivos do tipo `html` e `doc` no geral, apresentando boas taxas de *precision* e *recall*.

6. Conclusões

Os resultados obtidos neste trabalho nos permitem concluir que as estratégias não limitam a capacidade de detecção de similaridade das funções de pareamento aproximado por elas implementadas. Em alguns casos, mostramos que as estratégias fazem melhor uso dos resumos gerados, detectando mais similaridades do que as próprias ferramentas. Por fim, foi analisado o impacto do tipo de arquivo na capacidade de detecção das estratégias, onde se identificou uma dificuldade em encontrar arquivos do tipo texto com pequenos trechos de similaridades espalhados pelos arquivos. Além disto, verificamos um alto número de falsos positivos de arquivos pdf, devido aos *blocos comuns*, com conteúdo gerado por aplicação.

Referências

Breitinger, F. and Roussev, V. (2014). Automated evaluation of approximate matching algorithms on real data. *Digital Investigation*, 11:S10–S17.

Moia, V. H. G., Breitinger, F., and Henriques, M. A. A. (2020). The impact of excluding common blocks for approximate matching. *Computers & Security*, 89:101676.

Moia, V. H. G. and Henriques, M. A. A. (2017). Similarity digest search: A survey and comparative analysis of strategies to perform known file filtering using approximate matching. *Security and Communication Networks*, pages 1–17.

Roussev, V. (2011). An evaluation of forensic similarity hashes. *Digital investigation*, 8:34–41.

Winter, C., Schneider, M., and Yannikos, Y. (2013). F2s2: Fast forensic similarity search through indexing piecewise hash signatures. *Digital Investigation*, 10(4):361–371.